The Embedded I/O Company

TEWS TECHNOLOGIES

TPMC700-SW-82

Linux Device Driver

32(16) Digital Output PMC

Version 2.0.x

User Manual

Issue 2.0.0

June 2020

powerBridge Computer

Ehlbeek 15a
30938 Burgwedel
fon 05139-9980-0
fax 05139-9980-49

www.powerbridge.de
info@powerbridge.de

101 4058 0
1 4058 19
s.com

TEWS TECHNOLOGIES LLC
9190 Double Diamond Parkway,
Suite 127, Reno, NV 89521, USA
www.tews.com

Phone: +1 (775) 850 5830
Fax: +1 (775) 201 0347
e-mail: usasales@tews.com

TPMC700-SW-82

Linux Device Driver

32(16) Digital Output PMC

Supported Modules:
TPMC700

Issue	Description	Date
1.0.0	First Issue	December 1, 2006
1.0.1	General Revision	May 22, 2009
2.0.0	API implemented, Description of ioctl Interface removed	June 24, 2020

Table of Contents

1 Introduction

The TPMC700-SW-82 Linux device driver allows the operation of the TPMC700 PMC conforming to the Linux I/O system specification.

The TPMC700-SW-82 device driver supports the following features:

- writing a new output value
- enable and disable the output watchdog
- acknowledge watchdog errors

The TPMC700-SW-82 device driver supports the modules listed below:

TPMC700	32/16 Digital Outputs (24V, 0.5A) (High Side Switches)	(PMC)

To get more information about the features and use of TPMC700 devices it is recommended to read the manuals listed below.

TPMC700 User Manual

2 Installation

The directory TPMC700-SW-82 on the distribution media contains the following files:

TPMC700-SW-82-2.0.0.pdf	This manual in PDF format
TPMC700-SW-82-SRC.tar.gz	GZIP compressed archive with driver source code
Release.txt	Release information
ChangeLog.txt	Release history

The GZIP compressed archive TPMC700-SW-82-SRC.tar.gz contains the following files and directories:

Directory path 'tpmc700':

tpmc700.c	Driver source code
tpmc700def.h	Driver private include file
tpmc700.h	Driver public include file for application program
Makefile	Device driver make file
makenode	Script to create device nodes on the file system
COPYING	Copy of the GNU Public License (GPL)
api/tpmc700api.h	API include file
api/tpmc700api.c	API source file
include/tpxxxhwdep.c	Low level hardware access functions source file
include/tpxxxhwdep.h	Access functions header file
include/tpmodule.c	Driver independent library
include/tpmodule.h	Driver independent library header file
include/config.h	Driver independent library header file
example/tpmc700exa.c	Example application
example/Makefile	Example application make file

In order to perform an installation, extract all files of the archive TPMC700-SW-82-SRC.tar.gz to the desired target directory. The command 'tar -xzvf TPMC700-SW-82-SRC.tar.gz' will extract the files into the local directory.

2.1 Build and install the Device Driver

- Login as *root*
- Change to the target directory
- To create and install the driver in the module directory */lib/modules/<version>/misc* enter:

  # make install

- To update the device driver's module dependencies, enter:

  # depmod -aq

2.2 Uninstall the Device Driver

- Login as *root*
- Change to the target directory
- To remove the driver from the module directory */lib/modules/<version>/misc* enter:

  # make uninstall

2.3 Install Device Driver into the running Kernel

- To load the device driver into the running kernel, login as root and execute the following commands:

  # modprobe tpmc700drv

- After the first build or if you are using dynamic major device allocation it's necessary to create new device nodes on the file system. Please execute the script file *makenode* to do this. If your kernel has enabled a device file system (devfs or sysfs with udev) then you have to skip running the *makenode* script. Instead of creating device nodes from the script the driver itself takes creating and destroying of device nodes in its responsibility.

  # sh makenode

On success the device driver will create a minor device for each TPMC700 module found. The first TPMC700 module can be accessed with device node /dev/tpmc700_0, the second with device node /dev/tpmc700_1 and so on.

The assignment of device nodes to physical TPMC700 modules depends on the search order of the PCI bus driver.

2.4 Remove Device Driver from the running Kernel

- To remove the device driver from the running kernel login as root and execute the following command:

  # modprobe –r tpmc700drv

If your kernel has enabled devfs or sysfs (udev), all /dev/tpmc700_x nodes will be automatically removed from your file system after this.

Be sure that the driver isn't opened by any application program. If opened you will get the response "*tpmc700drv: Device or resource busy*" and the driver will still remain in the system until you close all opened files and execute *modprobe –r* again.

2.5 Change Major Device Number

This paragraph is only for Linux kernels without DEVFS installed. The TPMC700 device driver uses dynamic allocation of major device numbers per default. If this isn't suitable for the application it is possible to define a major number for the driver.

To change the major number edit the file tpmc700def.h, change the following symbol to appropriate value and enter `make install` to create a new driver.

TPMC700_MAJOR	Valid numbers are in range between 0 and 255. A value of 0 means dynamic number allocation.

Example:

```
#define TPMC700_MAJOR        122
```

Be sure that the desired major number isn't used by other drivers. Please check *\/proc/devices* to see which numbers are free.

3 API Documentation

3.1 General Functions

3.1.1 tpmc700Open

NAME

tpmc700Open – open a device

SYNOPSIS

```
TPMC700_HANDLE tpmc700Open
(
      char          *DeviceName
)
```

DESCRIPTION

Before I/O can be performed to a device, a device handle must be opened by a call to this function.

The tpmc700Open function can be called multiple times (e.g. in different tasks).

PARAMETERS

DeviceName

This parameter points to a null-terminated string that specifies the name of the device. The first TPMC700 device is named “/dev/tpmc700_0” the second device is named “/dev/tpmc700_1” and so on.

EXAMPLE

```
#include "tpmc700api.h"

TPMC700_HANDLE     hdl;

/*
** open the specified device
*/
hdl = tpmc700Open("/dev/tpmc700_0");
if (hdl == NULL)
{
     /* handle open error */
}
```

RETURNS

A device handle, or NULL if the function fails. An error code will be stored in *errno*.

ERROR CODES

The error codes are stored in *errno.*

The error code is a standard error code set by the I/O system.

3.1.2 tpmc700Close

NAME

tpmc700Close – close a device

SYNOPSIS

```
TPMC700_STATUS tpmc700Close
(
        TPMC700_HANDLE        hdl
)
```

DESCRIPTION

This function closes a previously opened device.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

EXAMPLE

```
#include "tpmc700api.h"

TPMC700_HANDLE     hdl;
TPMC700_STATUS     result;

/*
** close the device
*/
result = tpmc700Close(hdl);
if (result != TPMC700_OK)
{
    /* handle close error */
}
```

RETURNS

On success, TPMC700_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC700_ERR_INVALID_HANDLE	The specified device handle is invalid

3.1.3 tpmc700GetPciInfo

NAME

tpmc700GetPciInfo – get PCI information of the module

SYNOPSIS

```
TPMC700_STATUS tpmc700GetPciInfo
(
        TPMC700_HANDLE              hdl,
        TPMC700_PCIINFO_BUF         *pPciInfoBuf
)
```

DESCRIPTION

This function returns information about the module's PCI header as well as the PCI localization.

PARAMETERS

hdl

This argument specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

pPciInfoBuf

This argument is a pointer to the structure TPMC700_PCIINFO_BUF that receives information of the module PCI header.

```
typedef struct
{
        unsigned short      vendorId;
        unsigned short      deviceId;
        unsigned short      subSystemId;
        unsigned short      subSystemVendorId;
        int                 pciBusNo;
        int                 pciDevNo;
        int                 pciFuncNo;
} TPMC700_PCIINFO_BUF;
```

vendorId

PCI module vendor ID.

deviceId

PCI module device ID

subSystemId

PCI module sub system ID

subSystemVendorId

PCI module sub system vendor ID

pciBusNo

Number of the PCI bus, where the module resides.

pciDevNo

PCI device number

pciFuncNo

PCI function number

EXAMPLE

```
#include "tpmc700api.h"

TPMC700_HANDLE          hdl;
TPMC700_STATUS          result;
TPMC700_PCIINFO_BUF     pciInfoBuf;

/*
** get module PCI information
*/
result = tpmc700GetPciInfo( hdl, &pciInfoBuf );
if (result == TPMC700_OK)
{
     printf( "PCI Localization (Bus:Dev.Func): %d:%d.%d\n",
                    pciInfoBuf.pciBusNo,
                    pciInfoBuf.pciDevNo,
                    pciInfoBuf.pciFuncNo );
}
else
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC700_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC700_ERR_INVALID_HANDLE	The specified device handle is invalid
TPMC700_ERR_INVAL	Specified pointer is invalid.

3.2 Output Functions

3.2.1 tpmc700Write

NAME

tpmc700Write – Write Output Value

SYNOPSIS

```
TPMC700_STATUS tpmc700Write
(
        TPMC700_HANDLE              hdl,
        unsigned int                OutputValue
)
```

DESCRIPTION

This function writes the specified output value (32bit) to the specific module.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

OutputValue

This argument specifies the new output value. Bit 0 of the output word corresponds to the first output line, bit 1 corresponds to the second output line, and so on.

Bit 16 up to 32 will be ignored for TPMC700-x1 (16 output lines).

EXAMPLE

```
#include "tpmc700api.h"

TPMC700_HANDLE     hdl;
TPMC700_STATUS     result;

/*------------------------------
  Set output lines to 0x12345678
  ------------------------------*/
result = tpmc700Write(  hdl,
                        0x12345678 );
if (result != TPMC700_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC700_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC700_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC700_ERR_TIMEOUT	Watchdog Timeout error occurred.

3.2.2 tpmc700WriteMask

NAME

tpmc700WriteMask – Write Output Value with Bitmask

SYNOPSIS

```
TPMC700_STATUS tpmc700WriteMask
(
        TPMC700_HANDLE              hdl,
        unsigned int                OutputValue,
        unsigned int                Mask
)
```

DESCRIPTION

This function sets the output lines to the specified value. Only output lines specified by the bitmask are affected.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

OutputValue

This argument specifies the new output value. Bit 0 of the output word corresponds to the first output line, bit 1 corresponds to the second output line, and so on.

Bit 16 up to 32 will be ignored for TPMC700-x1 (16 output lines).

Mask

This parameter specifies a 32bit mask. '1' means that the corresponding bit in *OutputValue* will be updated. '0' bits will be left unchanged. Bit 0 corresponds to the first output line, bit 1 corresponds to the second output line and so on.

Bit 16 up to 32 will be ignored for TPMC700-x1 (16 output lines).

EXAMPLE

```
#include "tpmc700api.h"

TPMC700_HANDLE     hdl;
TPMC700_STATUS     result;
unsigned int       OutputValue;
unsigned int       Mask;

/*--------------------------------------------------
  Set output line 1 and 8 (bit 0 and bit 7), and
  clear output line 32 (bit 31)
  --------------------------------------------------*/
OutputValue    = (1 <<  7) | (1 << 0);
Mask           = (1 << 31) | (1 << 7) | (1 << 0);

result = tpmc700WriteMask(   hdl,
                             OutputValue,
                             Mask );
if (result != TPMC700_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC700_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC700_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC700_ERR_TIMEOUT	Watchdog Timeout error occurred.

3.2.3 tpmc700OutputLineSet

NAME

tpmc700OutputLineSet – Set the specific Output Line

SYNOPSIS

```
TPMC700_STATUS tpmc700OutputLineSet
(
        TPMC700_HANDLE          hdl,
        int                     OutputLine
)
```

DESCRIPTION

This function sets the specified output line to ‘1’.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

OutputLine

This argument specifies the output line number which shall be set. Valid values are 1 to 32. For TPMC700-x1 modules, values higher than 16 are ignored.

EXAMPLE

```
#include "tpmc700api.h"

TPMC700_HANDLE     hdl;
TPMC700_STATUS     result;

/*--------------------------------------------------
  Set output line 4
  --------------------------------------------------*/
result = tpmc700OutputLineSet( hdl, 4 );
if (result != TPMC700_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC700_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC700_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC700_ERR_TIMEOUT	Watchdog Timeout error occurred.
TPMC700_ERR_INVAL	Specified output line is invalid.

3.2.4 tpmc700OutputLineClear

NAME

tpmc700OutputLineClear – Clear the specific Output Line

SYNOPSIS

```
TPMC700_STATUS tpmc700OutputLineClear
(
        TPMC700_HANDLE          hdl,
        int                     OutputLine
)
```

DESCRIPTION

This function clears the specified output line to ‘0’.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

OutputLine

This argument specifies the output line number which shall be cleared. Valid values are 1 to 32. For TPMC700-x1 modules, values higher than 16 are ignored.

EXAMPLE

```
#include "tpmc700api.h"

TPMC700_HANDLE     hdl;
TPMC700_STATUS     result;

/*--------------------------------------------------
  Clear output line 32
  --------------------------------------------------*/
result = tpmc700OutputLineClear( hdl, 32 );
if (result != TPMC700_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC700_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC700_ERR_INVALID_HANDLE	The specified device handle is invalid.
TPMC700_ERR_TIMEOUT	Watchdog Timeout error occurred.
TPMC700_ERR_INVAL	Specified output line is invalid.

3.2.5 tpmc700OutputStatus

NAME

tpmc700OutputStatus – Read Status of Output Lines and Watchdog

SYNOPSIS

```
TPMC700_STATUS tpmc700OutputStatus
(
        TPMC700_HANDLE              hdl,
        unsigned int                *pOutputValue,
        unsigned int                *pWatchdogStatus
)
```

DESCRIPTION

This function reads the status of the output lines and also the watchdog facility.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

pOutputValue

This argument is a pointer to an *unsigned int* (32bit) value where the output line status is returned. Bit 0 of the output word corresponds to the first output line, bit 1 corresponds to the second output line, and so on. For TPMC700-x1, the upper 16bits shall be ignored.

pWatchdogStatus

This argument is a pointer to an *unsigned int* (32bit) value where the watchdog status is returned. The following values are possible:

Value	**Description**
TPMC700_WD_ENABLED	The Watchdog is enabled and active.
TPMC700_WD_DISABLED	The Watchdog is disabled.
TPMC700_WD_FAILURE	The Watchdog has recognized a failure and has disabled all output channels.

EXAMPLE

```
#include "tpmc700api.h"

TPMC700_HANDLE      hdl;
TPMC700_STATUS      result;
unsigned int        OutputValue;
unsigned int        WatchdogStatus;

/*--------------------------------------------------
  Read output status
  --------------------------------------------------*/
result = tpmc700OutputStatus(    hdl,
                                 &OutputValue,
                                 &WatchdogStatus );
if (result == TPMC700_OK)
{
     if (WatchdogStatus != TPMC700_WD_FAILURE)
     {
          printf("Output Status: 0x%08X\n", OutputValue);
     }
     else
     {
          printf("Output disabled by Watchdog\n");
     }
}
else
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC700_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC700_ERR_INVALID_HANDLE	The specified device handle is invalid.

3.3 Watchdog Functions

3.3.1 tpmc700WatchdogEnable

NAME

tpmc700WatchdogEnable – Enable Output Watchdog

SYNOPSIS

```
TPMC700_STATUS tpmc700WatchdogEnable
(
      TPMC700_HANDLE              hdl
)
```

DESCRIPTION

This function enables the watchdog timer for the output lines. The watchdog function is activated after the next write operation to the device. Please remember that if the watchdog is enabled and no write access occurs within 120 ms, all outputs go into the OFF state. To unlock the output register and leave the OFF state the function *tpmc700WatchdogReset* must be executed.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

EXAMPLE

```
#include "tpmc700api.h"

TPMC700_HANDLE     hdl;
TPMC700_STATUS     result;

/*-----------------
  Enable Watchdog
  -----------------*/
result = tpmc700WatchdogEnable( hdl );
if (result != TPMC700_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC700_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC700_ERR_INVALID_HANDLE	The specified device handle is invalid.

3.3.2 tpmc700WatchdogDisable

NAME

tpmc700WatchdogDisable – Disable Output Watchdog

SYNOPSIS

```
TPMC700_STATUS tpmc700WatchdogDisable
(
        TPMC700_HANDLE                  hdl
)
```

DESCRIPTION

This function disables the watchdog timer for the output lines.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

EXAMPLE

```
#include "tpmc700api.h"

TPMC700_HANDLE     hdl;
TPMC700_STATUS     result;

/*------------------
  Disable Watchdog
  ------------------*/
result = tpmc700WatchdogDisable( hdl );
if (result != TPMC700_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC700_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC700_ERR_INVALID_HANDLE	The specified device handle is invalid.

3.3.3 tpmc700WatchdogReset

NAME

tpmc700WatchdogReset – Reset Output Watchdog Error

SYNOPSIS

```
TPMC700_STATUS tpmc700WatchdogReset
(
      TPMC700_HANDLE            hdl
)
```

DESCRIPTION

This function resets the watchdog status and clears an occurred error.

PARAMETERS

hdl

This value specifies the device handle to the hardware module retrieved by a call to the corresponding open-function.

EXAMPLE

```
#include "tpmc700api.h"

TPMC700_HANDLE     hdl;
TPMC700_STATUS     result;

/*-----------------
  Reset Watchdog
  -----------------*/
result = tpmc700WatchdogReset( hdl );
if (result != TPMC700_OK)
{
     /* handle error */
}
```

RETURN VALUE

On success, TPMC700_OK is returned. In the case of an error, the appropriate error code is returned by the function.

ERROR CODES

Error Code	Description
TPMC700_ERR_INVALID_HANDLE	The specified device handle is invalid.

4 Diagnostic

If the TPMC700 does not work properly it is helpful to get some status information from the driver respective kernel.

The Linux */proc* file system provides information about kernel, resources, drivers, devices and so on. The following screen dumps displays information of a correct running TPMC700 device driver (see also the proc man pages).

```
# lspci -v
  …
04:01.0 Signal processing controller: TEWS Technologies GmbH Device 02bc
(rev 01)
        Subsystem: TEWS Technologies GmbH Device 000a
        Flags: medium devsel
        Memory at feb9fc00 (32-bit, non-prefetchable) [size=128]
        I/O ports at e880 [size=128]
        Memory at feb9f800 (32-bit, non-prefetchable) [size=16]
        Kernel driver in use: TEWS TECHNOLOGIES - TPMC700 Digital Output -
        Kernel modules: tpmc700drv
  …
```

```
# cat /proc/devices
Character devices:
  1 mem
  …
247 tpmc700drv
  …
```

```
# cat /proc/ioports
0000-0cf7 : PCI Bus 0000:00
  0000-001f : dma1
  0020-0021 : pic1
  …
  e000-efff : PCI Bus 0000:04
    e800-e87f : 0000:04:02.0
    e880-e8ff : 0000:04:01.0
    ec00-ec3f : 0000:04:00.0
  …
```

```
# cat /proc/iomem
00000000-00000fff : Reserved
00001000-0009fbff : System RAM
0009fc00-0009ffff : Reserved
…
  feb00000-febfffff : PCI Bus 0000:04
    feb9f000-feb9f01f : 0000:04:02.0
    feb9f400-feb9f47f : 0000:04:02.0
    feb9f800-feb9f80f : 0000:04:01.0
      feb9f800-feb9f80f : TPMC700
    feb9fc00-feb9fc7f : 0000:04:01.0
    feba0000-febbffff : 0000:04:00.0
    febc0000-febdffff : 0000:04:00.0
…
```